# BEBR

**FACULTY WORKING PAPER NO. 1207**

## Applications of Artificial Intelligence to Planning and Scheduling in Flexible Manufacturing

*Michael J. P. Shaw*

College of Commerce and Business Administration
Bureau of Economic and Business Research
University of Illinois, Urbana-Champaign

# BEBR

FACULTY WORKING PAPER NO. 1207

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

November, 1985

Applications of Artificial Intelligence to Planning and Scheduling in Flexible Manufacturing

Michael J. P. Shaw, Assistant Professor
Department of Business Administration

Research supported in part by the Office for Information Management under a project funded by IBM, the Research Board of UIUC, and a gift from AMOCO Foundation.

ABSTRACT

In this paper we investigate recent developments of artificial intelligence techniques, and address the application of these techniques to the planning and scheduling of flexible manufacturing systems. We have developed a nonlinear planning system, currently implemented in LISP, to demonstrate the knowledge-based organization and address the important issues for AI based planning and scheduling. Of particular importance are the issues of knowledge representation, resource coordination, temporal reasoning, and scheduling efficiency.

# I. INTRODUCTION

Recent advances in artificial intelligence (AI) have had an enormous impact on information processing technologies used in the discrete-part manufacturing industry. Recent successes of AI knowledge-based systems developments have created an opportunity for much needed breakthroughs in the planning, scheduling, and control of flexible manufacturing systems. Indeed, there is a widespread feeling that the development of AI systems has become the new frontier of practical applications of computers.

Flexible automation--automation that can handle a large and constantly changing variety of produced items--has played an increasingly important role in the efforts to improve the productivity of the manufacturing industry (Hutchingson [1984], Merchang [1983]). The recent progress in computer technologies has accelerated the realization of flexible automation. The use of computers in manufacturing, such as the numerically controlled (NC) machines, adds programmability and thus versatility into manufacturing systems. More important, computers also provide on-line execution of manufacturing planning and decision making. These two capabilities, computerized control and on-line planning, are integrated into a well-orchestrated, automated manufacturing system that can produce wide-ranging items efficiently.

The implementation of computerized machine centers, robots, and other flexible manufacturing technologies increase the number of alternative ways a product can be produced. At the same time, the complexity of planning and scheduling has increased due to the multiprocessing environment and the dynamically changing states.

Furthermore, as a result of more efficient machine tools and robots, the lead-time allowed for planning and scheduling is shortened, thus calling for good, real-time performance on the decision-making unit. To meet these performance requirements, we need to explore a scheduling approach more capable than the traditional approaches; it should have the following capabilities:

1. Providing on-line decision support.
2. Scheduling operations dynamically.
3. Coordinating manufacturing resources.
4. Synchronizing processes for different jobs.
5. Monitoring the plan execution.

We will present a knowledge-based approach to achieve these planning and scheduling requirements.

The knowledge-based organization provides several advantages in problem-solving, e.g., the flexible control structure, the extendable and transparent knowledge representation scheme, and the reasoning ability. These systems are called knowledge-based primarily because their performance depends critically on the use of facts and heuristics; thus, for implementing a successful knowledge-based system, it is always important to use an effective representation scheme to capture the facts of the problem. At the same time, the system should be equipped with procedural knowledge, providing information about feasible actions to take in a given situation. The system should be able to reason with judgmental knowledge as well as with formal, established knowledge. As such, specific knowledge of

the problem domain is coupled with general problem solving knowledge to provide expert-level analysis of difficult situations.

The use of knowledge-based systems for intelligent problem-solving can trace back to Newell and Simon's work on general problem solver (GPS), in which they attempted to build a computer program to model human problem-solving processes (Newell and Simon [1963]). GPS incorporated an organization where procedural knowledge, stored in a production system (Davis [1977]), is represented by IF-THEN rules. A database, serving as the short-term memory, contains the facts collected. Modus ponens is the primary rule of inference by which a system adds new facts to a growing data-base. Work on knowledge-based expert systems began to flower in the 1970s, especially in medical problem areas, exemplified by the MYCIN project (Shortliff [1976]), where a large part of the knowledge is the body of judgemental heuristics that human experts use in solving hard problems.

In an effort to improve manufacturing productivity, research work applying AI knowledge-based technology to manufacturing operations began to emerge. Knowledge-based systems have been developed to deal with various manufacturing applications. Descotte and Latcome [1982] and Chang and Nau [1981] used expert systems for manufacturing process planning which concerns the selection of processing steps for machining mechanical parts. Fox [1983] developed a knowledge-based system, ISIS, for large scale job-shop scheduling. R1 (McDermott [1982]) is an expert for configuring computers built for Digital Equipment Corporation which, as a result of R1's success, started an effort to apply knowledge-based technology to their manufacturing

operations. For example, ISA is a knowledge-based system that provides the capability of scheduling customer system orders against the current planned material allocation; IMACS is an intelligent system for assisting managers in paperwork management, capacity planning, and inventory management (O'Connor [1984]).

In the remainder of this paper, Section II explores the knowledge-based organization for automatic planning. Section III characterizes the environment and identifies the planning and scheduling issues in flexible manufacturing systems. In Section IV, a prototype of the scheduling system is used to illustrate the fourstep nonlinear planning scheme we have developed. Section V further discusses such special scheduling capabilities as dynamic scheduling and distributed scheduling.

## II. KNOWLEDGE-BASED AUTOMATIC PLANNING

### The Framework

Knowledge-based systems treat knowledge on three levels: data, knowledge base, and control. By contrast, conventional programs treat knowledge on only two levels: data and program. At the data level a knowledge-based system stores declarative knowledge about the goals, the current situation of the world, and the semifinished plan. At the knowledge-base level is stored the domain-specific, procedural knowledge. This knowledge models the actions of the world, and it is often represented by production rules or operators. Finally, at the control level the knowledge about the strategy of plan construction is stored; this is the control knowledge indicating how to select operators, and when to apply them.

For our application because planning involves exploration of alternative sequences of actions, a symbolic model of the real world, the "world model," represents the environment as the plans evolve. For any given planning problem, the initial condition and the stated goal condition are both treated as instances in the world model. The generation function of a planning system, then, is to construct a course of action that transforms one state of the world model, which contains an initial condition, to a state which satisfies the goal condition. Thus the planning system we have developed has three basic components:

1) A world model. Represented as declarative knowledge at the data level, this world model contains relevant descriptions about the environment. This knowledge is usually about the properties of domain objects (e.g., whether a machine is idle) or about the relationships among the objects (e.g., whether a part is at a certain machine). The most prevalent knowledge representation for the world model is the use of first-order predicate calculus to describe properties, functions or relations of objects (Nilson [1980]). In the planning system, corresponding to every possible situation of the environment, the world model is represented by a conjunction of the predicate formulas that hold true in that particular situation. This conjunction of instances of predicate formulas defines a "state" corresponding to a situation in the real-world environment.

Some of the properties of objects and relationships among objects may not change over time. For example, the capabilities of a machine

or the connectivity between two machines do not change with the application of operators; these properties are called the invariant properties of domain objects. This separation of state-changing knowledge and invariant knowledge is analogous to the reasoning process of human experts; the knowledge that changes with actions is stored in a working memory, whereas the knowledge that remains unchanged is stored in the long-term memory. The same idea is applied in representing the world model.

The invariant properties of domain objects can be structuredly represented by semantic networks or frame-based systems to take advantage of their greater efficiency. In these representations, all the relevant information is collected together and properties can be inherited, so that accessing and manipulating the information can be facilitated. Further, constraints can be specified to regulate the possible values of variable instantiations reducing the feasible domain for the solution-searching procedures.

2) An action model. This action model, represented at the domain-specific knowledge-base level, formalizes the description of applicabilities of each action and its impact on the world model when applied. An action is represented by a transformation from one state of the world model to another state; each action is represented as an operator. Using the pattern-directed representation, each operator consists of two components: preconditions and postconditions. The preconditions are represented by a predicate calculus expression, if the expression--the pattern--is matched by the state descriptions in the world model, the operator becomes applicable. The postconditions

define the literals that are added or deleted by the application or operators.

When using the action model, the "frame problem" must be considered. The frame problem is the problem of specifying which conditions in a state description should change and which should not, whereas almost all conditions which hold true in a given state continue to hold true after an action has been performed. Based on the assumption of the STRIPS system, the action model assumes that all conditions that are not indicated to be changed by the operator remain the same; that is, all the changes in the world are accounted for by the postconditions of the applied operator. In our view, this assumption is valid for the manufacturing environment where the effects of the manufacturing environment can be clearly defined. A counter example of this assumption is the ill-designed block world, where stacking a block onto a stack of blocks, for instance, might topple the whole stack of blocks. Such a world is difficult for planning because of the unpredictability of action effects.

While the operators are used to describe the actions relevant to the problem domain, these could be other rules which provide good judgment of actions to take when specific situations arise. These are judgmental or empirical knowledge in the knowledge base, usually acquired from human experts. This type of knowledge is excluded from robot planning systems, where actions are well defined and situations are tightly controlled. In the general computer integrated manufacturing environment, however, the introduction of these judgmental rules may be necessary.

3) An inference engine, as the control unit. The major control decision, exercised by the inference engine, is concerned with the selection of action sequences which are based on the current state of the world model and the decision of what action can lead to the goal state. Since the generation of an action sequence for a plan typically involves extensive searches among alternative actions, the inference engine produces a search tree along the plan-generation process. The goal description is at the root node, with instances of operators defining branches, and the intermediate state defines the nodes. The plan generation is then equivalent to a graph search procedure; standard search strategies such as best-first or back-tracking can be used to guide the search. Moreover, heuristic rules are sometimes used to facilitate the search of operators. An example of using heuristic rules is the "means-ends analysis." When an operator is applied with this algorithm, the difference between the new state and the goal state is determined and the operator that can best reduce the difference is chosen. This "searching cycle" continues until the goal is satisfied in the new state. If the chosen operator is not applicable, its preconditions are established as a new intermediate subgoal. The algorithm is then applied in a recursive fashion to achieve the subgoal by the same searching cycles.

To illustrate the strategies involved in means-ends analysis, the search algorithm for generating a linearly sequenced plan is shown in the following procedures:

```
Procedure OPERATOR-SEARCH(G)

Input:
    S:  the initial state
    G:  the goal state

Output:
    Plan:  the resulting linearly sequenced plan

Begin

    1) Plan := Nil

    2) Until S matches G Do Begin

    3)   g := G - S /* unsatisfied goal */

    4)   op := operators whose add list contains a literal that
             matches g
             /* applicable set */

    5)   p* := nondeterministically select an operator from op

    6)   Plan := concatenate (plan,p*)
             /* plan contains all operators used so far */

    7)   q := precondition formula of appropriate instance of p*
             /* subgoal */

    8)   Call OPERATOR-SEARCH(q)
             /* node expansion */

    9)   S := result of applying p* to S
         End

End
```

This procedure is similar to the plan generation procedure used in STRIPS (Nilson [1980]). Using the "means-ends analysis" planning procedure, those components of G unmatched by S are treated as the difference between the goal state and the current state, denoted by g; operators whose add list contain any literal in g are considered relevant to reducing the difference and therefore are applicable. Many operators may be applicable, but only one can be selected nondeterministically.

For large planning systems, the number of alternatives to search at each choice point can grow very fast with the size of the problem. There are two basic methods for overcoming the combinatorial explosion associated with the search in extremely complex planning problems. These two methods are:

(1) to search the space more efficiently, or

(2) to transfer the search space into smaller manageable chunks that can be searched efficiently.

The planning strategies based on the first method include heuristic search and constraint satisfaction; the strategies based on the second method include hierarchical planning, problem reduction, divide and conquer, and least commitment.

## III. THE PLANNING AND SCHEDULING IN FLEXIBLE MANUFACTURING

As a flexible manufacturing system, a manufacturing cell is usually a modular unit in a computer-integrated manufacturing system (Cutkosky [1983], Shaw and Whinston [1984], [1985]). A typical manufacturing cell has several computer-controlled machines and robots, with an automatic handling system transporting parts between machines. Such integrated systems are characterized by their ability to make different parts and to perform a wide range of operations. The necessary decision-making regarding the assignment of manufacturing resources (e.g., machines, robots, fixtures, tools) consists primarily of two subprocesses: process selection and scheduling. Process planning aims at matching the machinery operations to functional requirements of the pacts; scheduling seeks to assign necessary resources to each operation efficiently.

The purpose of process planning is to produce a process plan, a partially ordered network of operations, for producing a part, given its engineering design. The process plan specifies the machining operations, the precedence ordering, tool requirements, and the surface finishing. The process-planning problem is a complicated problem that involves mostly judgmental decisions. For example, there can be a rule dictating the ordering of such operations as reaming and drilling: a reaming operation is always preceded by a drilling operation but not vice versa. Expert system technologies seem ideal for this type of decision making. For example, Chang and Nau [1981] used a frame-based expert system to solve the process selection problem.

The scheduling problem may be stated thus: A certain number, say n, parts are assigned to the manufacturing cell; each part requires a given set of linearly sequenced operations to be performed by the m machines. Since the machines have varying efficiencies for different operations, each part is routed among the machines so that every operation it needs is performed by the most appropriate machine available. The parameters associated with a job include the release time, the processing time, its due date, and the operations involved.

The objective of the problem is to schedule the n parts concurrently by developing a schedule for each part traveling among the machines; the makespan--the duration taken for completing all the required operations--should be minimized, while avoiding any conflicts arising from assigning parts to busy machines. The scheduling problem in the flexible manufacturing cell has these characteristics:

(1) Jobs consist of partially ordered operation sequences (as a result of process planning);

(2) A given operation can be performed on several alternative machines with different processing durations; and

(3) Each machine, while capable of performing a variety of operations, can execute only one operation at a time.

This scheduling problem can be formulated as an integer programming problem which captures all these characteristics (Shaw and Whinston [1985]).

The STRIPS system, an early planning system developed in artificial intelligence (Fikes, Hart, and Nilsson [1971], [1972]), used an inference procedure to construct plans that can guide the robot's motions in accomplishing specified goals. For systems such as STRIPS, the resulting plans are linearly sequenced, with strict precedence ordering between the selected actions. This is the simplest planning method--commonly referred to as linear planning. In real-world planning, however, there are usually conjunctive goals to be planned. (In the manufacturing world, each goal corresponds to a product.) Planning for conjunctive goals is referred to as nonlinear planning because the resulting plans are partially ordered. Nonlinear planning systems seek to break a problem into subproblems, using the so-called divide-and-conquer strategy. If the subproblems are independent of each other, then, in principle, the plan-generation processes for subgoals can be parallel. We call two segments of a plan parallel if there is no precedence relationship between them. Parallelism is considered beneficial for the planning process because of the resulting efficiency, which is the highest when subproblems are completely solved in parallel. In most of the cases, however, the actions of one plan interacts with the actions of other plans causing "conflicts"

between these linearly sequenced plans and disrupting the correctness of the plan. Nonlinear planning systems take these interactions into account and take measures to avoid conflicts between parallel actions (Sacerdoti [1977], Tate [1977]).

Vere [25] applied the nonlinear planning approach to scheduling space shuttles by a dynamic control scheme. In it, the time of the activities is described formally as a parameter in the planning system, and the goals are posted as functions of their ending times. Wilkins ([1982], [1984]) developed a domain-independent planning system featuring explicit resource coordination. For such planning domains as those in the flexible manufacturing system, where the detection of conflicts in resource assignment is essential, Wilkins' method can be more efficient than ordinary planning methods. Fox ([1982], [1983]) developed a planning and scheduling system by a constraint-directed approach for a large manufacturing system. In it, the key part of the search for machine schedules is the application of constraints to reduce the search space, thus speeding up the planning process. Although his target system is a large-scale job shop, the environment actually resembles that of a flexible manufacturing system since it is automated and computer-controlled. An alternative approach can be found in Bullers et al. [1980], where they apply problem reduction to perform planning and control in the manufacturing domain. Predicate logic and theorem proving techniques are used in deriving manufacturing steps in the dynamic environment.

The planning method we have developed (Shaw and Whinston [1985]) for the computer integrated manufacturing environment has characteristics

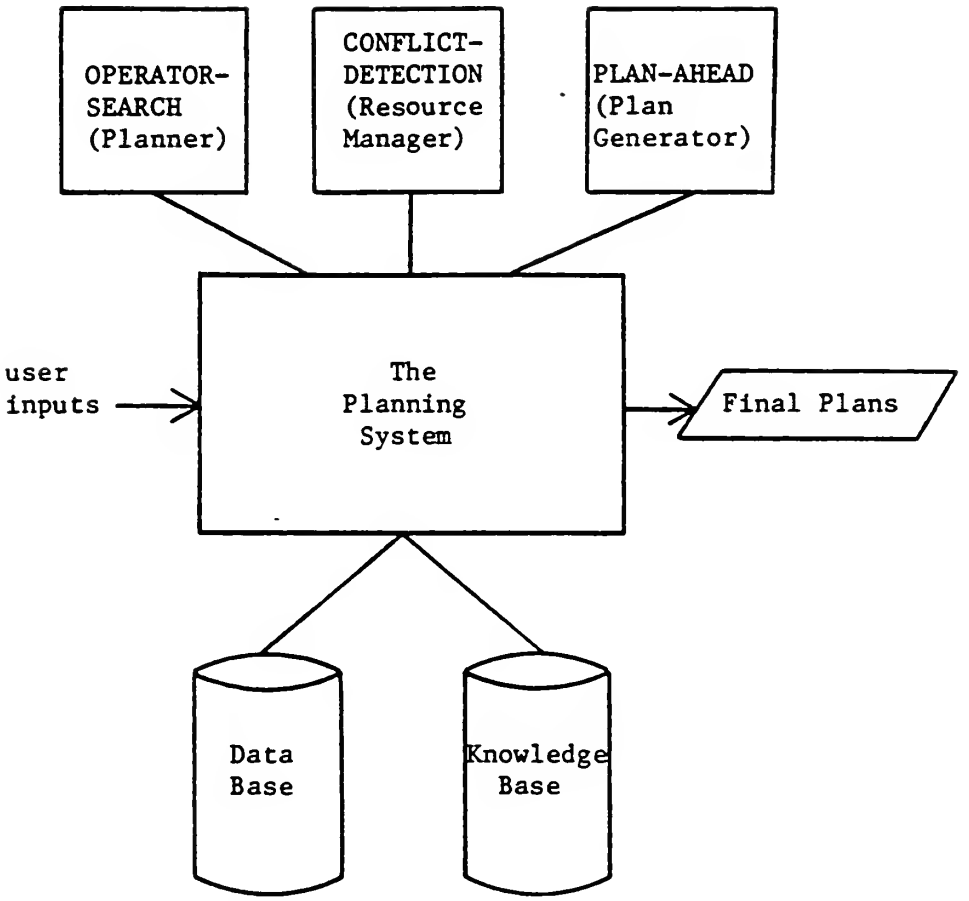
resembling those of the systems already discussed. Based on the nonlinear planning methods used in NOAH and DCOMP, our method begins by constructing a linear plan for each subgoal. As with DEVISER (Vere [1983]), our method uses the duration as an important description in the planning, especially in deciding precedence constraints. However, by emphasizing resources used by activities, the resulting plan network is more expensive for managerial analysis, e.g., sensitivity analysis like "what if a certain resource is not available."

To minimize the total time taken by the resulting plan, we use a plan generator to decide the best ordering among the activities. The plan generator runs the activities in the way they are arranged by the linear planner, with each operator scheduled into an event-list. An operator is blocked in a queue if the resource it requests is occupied; thus, a precedence constraint must be established in the plan between the operator currently using the resource and the operator which is blocked; the former is, then, the predecessor, the latter is the successor. The partial ordering thus constructed will minimize total duration of the plan.

## IV. A SCHEDULING SYSTEM

For the flexible manufacturing environment, Shaw and Whinston [1985] used a scheduling program, based on the nonlinear planning approach, to derive the desired production plans within each cell. This program is written in LISP on VAX 11/780, and has been applied to solving the scheduling problem in a manufacturing cell. The organization of the program is shown in Figure 1.

FIGURE 1


OPERATOR-SEARCH (Planner)
CONFLICT-DETECTION (Resource Manager)
PLAN-AHEAD (Plan Generator)
user inputs
The Planning System
Final Plans
Data Base
Knowledge Base

Using the concepts from nonlinear planning, an n-part-m-machine scheduling problem can be decomposed into n subproblems, with each subproblem defined as the routing of one part. The nonlinear planning method already discussed can thus be utilized to generate a plan for the n subproblems; the primary "interactions" between these subproblems are their sharing of the m machines. The objectives of the scheduling problem--to minimize makespan and to avoid conflicting assignments--can be translated into the criteria for the plan-generation problem: to maximize the parallelism and to avoid harmful interactions among the subplans.

The flexible manufacturing environment is symbolically represented by predicates in the database, as shown in Table 1. Primitive actions of the manufacturing processes are represented by operators, examples of which are shown in Figure 2 and Table 2.

Within such a planning framework, the manufacturing process corresponding to each job is modeled by state-changing transformations. Since the system usually works on several different jobs at once, proper coordination is essential. If the manufacturing processes for different jobs are independent, then, in principle, they can be executed in parallel. In reality, however, workpieces usually have to share machines, tools, and other resources. Therefore, the manufacturing processes operating on them must interact. The idea is to coordinate the planned processes operating on them must interact. The idea is to coordinate the planned activities for each job so that each manufacturing operation is performed by the most capable machine available, thereby making efficient use of the machines.

FIGURE 2

TRANSFER(M,M',PT,t): Transfer part PT from machine M to machine M' at time t.

Precondition: FINISH-OP(M,OP,PT,t)

DIFFERENT (M,M')

PT-NEXTOP(OP,OP',PT)

MACH-OP(M',OP')

IDLE(M',t)

Add-list: MACH-PT(M',OP',PT,t)

IDLE(M,t)

Delete-list: FINISH-OP(M,OP,PT,t)

IDLE(M',t)

Resource: M'

Duration: 2

EXECUTE(M,OP,PT,t): Execute operation OP on part PT on machine M at time t.

Precondition: MACH-PT(M,OP,PT,t)

TOOL(M,OP,t)

Add-list: FINISH-OP(M,OP,PT,t+$\delta$t)

Delete-list: MACH-PT(M,OP,PT,t)

Resource: M

Duration: $\delta$t

TABLE 1

IDLE(M,t): Machine M is idle at time t

MACHPT(M,OP,PT,t): Machine M begins operation OP on part PT at time t

FINISH-OP(M,OP,PT,t): Machine M completes operation OP on part PT at time t

DIFFERENT(M,M'): Machine M is a machine different from machine M'

MACH-OP(M,OP): Machine M is capable of performing operation OP

PT-FIRST-OP(OP,PT): Operation OP is the first operation on part PT

PT-NEXTOP(OP,OP',PT): Operation OP' should be performed on part PT immediately after operation OP

DONE(PT,t): All operations on part PT are completed at time t

TOOL(M,OP,t): The tool for operation OP is available to the machine M at time t

TABLE 2

| | |
|---|---|
| TRANSFER(M,M',PT,t): | Transfer part PT from machine M to machine M' at time t |
| NEXTOP(M,OP,OP',PT,t): | Perform operation OP' on part PT following operation OP on the same machine M |
| UNLOAD(M,DOCK,PT,t): | Unload part PT from machine M onto the unloading dock DOCK at time t |
| EXIT(PT,t): | Part PT which is unloaded at the unloading dock DOCK at time t |
| EXECUTE(M,OP,PT,t): | Execute operation OP on part PT on machine M at time t |
| ENTER(PT,t): | Part PT enters the system at time t |

Time is an important parameter for the planning system, and it must include not only the time span of the plan but also the time at which each activity occurs. Usually, the objective for such planning is to minimize the total time taken for completing the jobs (Baker [1974]). To take this objective into account, the planning system represents time explicitly in the knowledge base and uses sensitivity analysis to ensure that due dates are satisfied. Finally, the planning system considers alternative operations and revises existing plans if any bottlenecks are detected. The planning scheme is composed of four steps.

Step 1. Generate a linearly-sequenced plan for each task.

Step 2. Identify problematic interactions between the planning steps.

Step 3. Use precedence constraints to avoid conflicts.

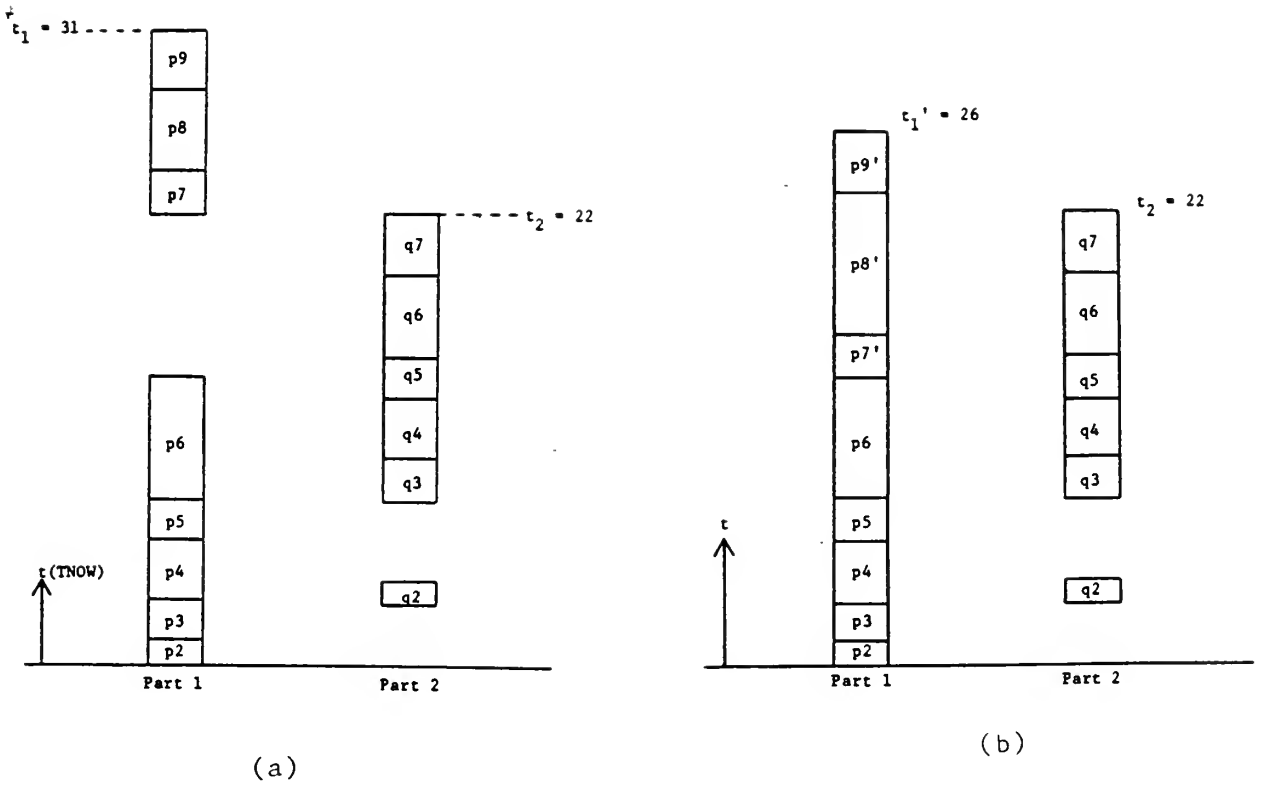
Step 4. Identify alternative planning steps to improve the performance.

An example of the partially ordered plan resulting from the scheme, represented by a Gannt chart, is shown in Figure 3, where each of the nodes represents a manufacturing planning operator.

In step 1 of the planning scheme, the inference engine calls upon a backward chaining procedure to search for the best planning steps in constructing the manufacturing plan for each part. In our program, this is carried out by a procedure, called OPERATOR-SEARCH, based on "means-ends analysis" heuristic (Nilson [1980]).

In each intermediate step of the planning, the inference engine must search through the rule-base to select the best operation to apply. Such a searching decision is exemplified by the node expansion

FIGURE 3


t1 = 31
p9
p8
p7
p6
p5
p4
p3
p2
t(TNOW)
Part 1
t2 = 22
q7
q6
q5
q4
q3
q2
Part 2
(a)
t1' = 26
p9'
p8'
p7'
p6
p5
p4
p3
p2
t
Part 1
t2 = 22
q7
q6
q5
q4
q3
q2
Part 2
(b)

shown in Figure 4. This searching step is very important for the total efficiency of plan generation, and the selection decision is usually aided by additional information represented in two forms: heuristics and constraints. Both types of knowledge are commonly used in AI problem-solving to facilitate the searching of the best operators. In the scheduling application, the heuristic function we use is calculated by

$$f(n) = (\text{estimated processing time for the pending operation}) + (\text{estimated total processing time for the remaining operations})$$

Constraints provide another major source of information for guiding searching. Examples of constraint knowledge include: deadline, precedence ordering, physical incompatability, and mutual exclusion. In our scheduling program, we have adopted an integration of heuristic and constraint knowledge. The shortest processing time heuristic is used to select operators, while latest finish time, mutual exclusion of resources, and physical constraints are applied at the same time. The search tree generated by this step, as shown in Figure 5, results in a set of linearly ordered planning operators.

A plan-generation procedure that executes steps 2 and 3 of nonlinear planning for conflict detection and resolution dynamically decides the precedence relationship between two conflicting actions. The underlying principle--based on the least commitment strategy--is not to impose any precedence constraint unless it is absolutely necessary, so that the parallelism among the subplans is maximized.

FIGURE 4

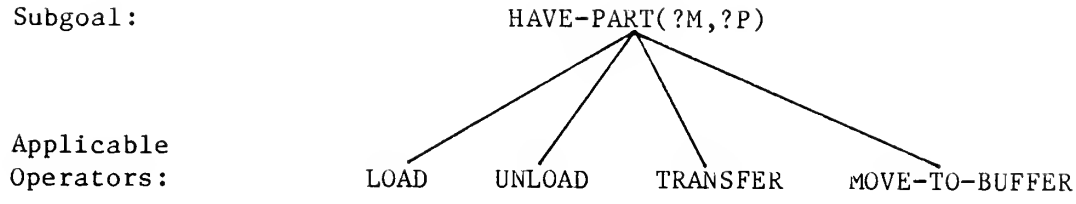
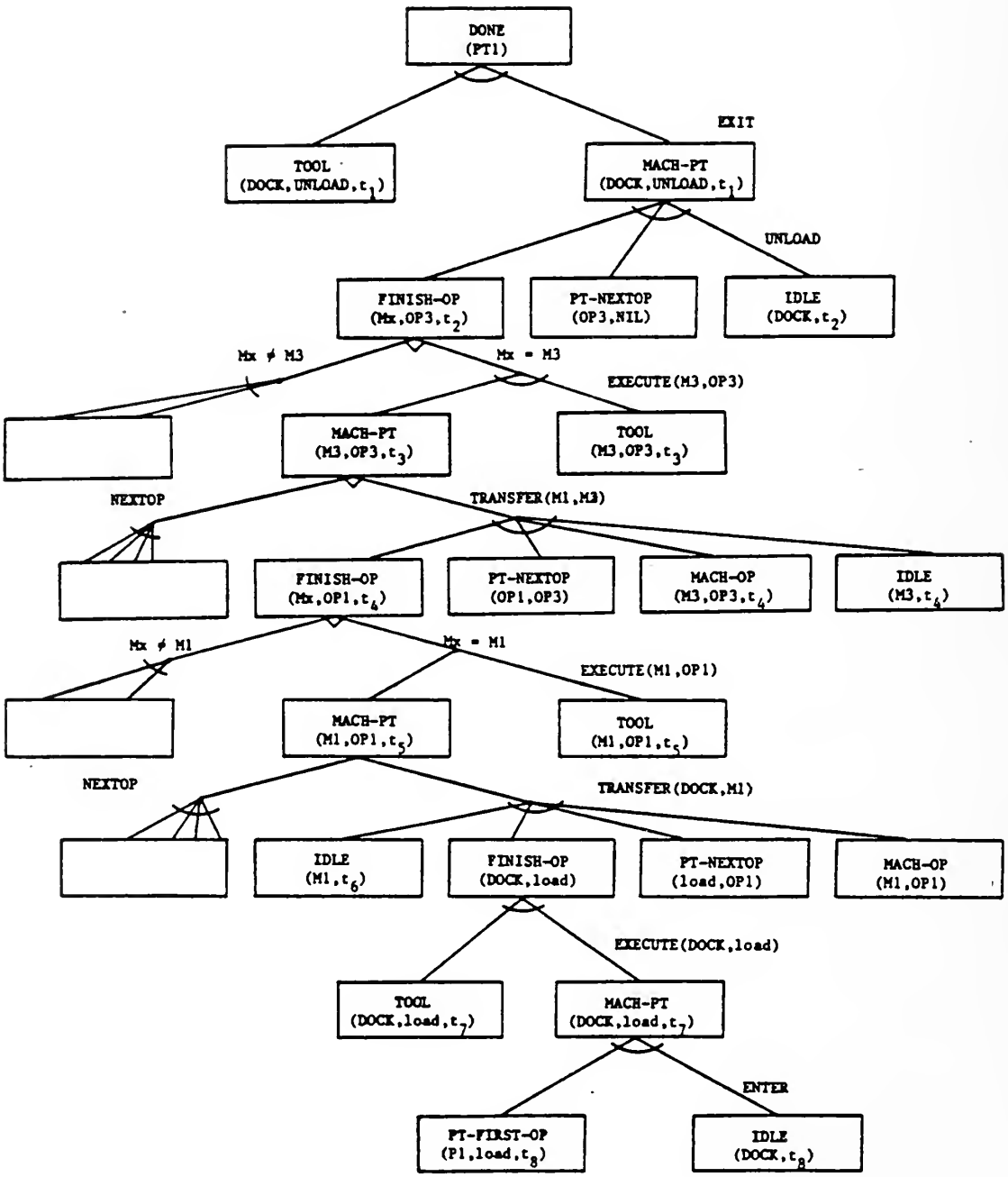

Subgoal:
HAVE-PART(?M,?P)
Applicable
Operators:
LOAD
UNLOAD
TRANSFER
MOVE-TO-BUFFER

FIGURE 5


DONE
(PT1)
EXIT
TOOL
(DOCK,UNLOAD,$t_1$)
MACH-PT
(DOCK,UNLOAD,$t_1$)
UNLOAD
FINISH-OP
(Mx,OP3,$t_2$)
PT-NEXTOP
(OP3,NIL)
IDLE
(DOCK,$t_2$)
Mx ≠ M3
Mx = M3
EXECUTE(M3,OP3)
MACH-PT
(M3,OP3,$t_3$)
TOOL
(M3,OP3,$t_3$)
NEXTOP
TRANSFER(M1,M3)
FINISH-OP
(Mx,OP1,$t_4$)
PT-NEXTOP
(OP1,OP3)
MACH-OP
(M3,OP3,$t_4$)
IDLE
(M3,$t_4$)
Mx ≠ M1
Mx = M1
EXECUTE(M1,OP1)
MACH-PT
(M1,OP1,$t_5$)
TOOL
(M1,OP1,$t_5$)
NEXTOP
TRANSFER(DOCK,M1)
IDLE
(M1,$t_6$)
FINISH-OP
(DOCK,load)
PT-NEXTOP
(load,OP1)
MACH-OP
(M1,OP1)
EXECUTE(DOCK,load)
TOOL
(DOCK,load,$t_7$)
MACH-PT
(DOCK,load,$t_7$)
ENTER
PT-FIRST-OP
(P1,load,$t_8$)
IDLE
(DOCK,$t_8$)

Information about resources and duration of actions is crucial to the inference engine in making these decisions.

Step 4 of the nonlinear planning scheme employs a method that reassigns waiting jobs to alternative resources so as to achieve better utilization and performance as much as possible. This procedure, called Plan-Revision, can be described as follows.

Resource Coordination

After a linear plan is constructed for each subgoal, the next step is to identify problematic interactions between parallel actions. The primary cause of such interactions is the potential conflicts in using resources. There are two possible approaches in this step: (1) a "critic mechanism," or (2) a "reasoning about resources" scheme. Let us consider each in turn.

The critic mechanism comes from the method used in NOAH (Sacerdoti [1977]) and DCOMP (Nilson [1980]). In these, the interaction-detection mechanism--called the "critic"--of the planning system identifies potentially harmful interactions between planning steps by checking the effects of the operators involved. If the preconditions of one operator are deleted by another, earlier operator, these preconditions must be added back by yet another operator, which will come between the two original operators. Thus, to test for potential conflicts facing an operator, two kinds of information are crucial: those operators in the plan that can delete its preconditions, and those operators in the plan that can result in its preconditions; the former are recorded in an "adder list," the latter are recorded in a

"deleter list." These two lists are parts of a table kept by the planner, referred to as the table of multiple effects (TOME).

In the flexible manufacturing environment, the major cause of conflicts is the sharing of resources between jobs. The preceding approach using TOME is one way to construct the resource sharing constraint. Alternatively, the planning system can utilize the resource information explicitly in coordinating activities and synthesizing subplans. In a broader sense, a "resource" can be defined as an object used by the action during its application. A resource cannot be shared by more than one action. The declaration of resource information by an action imposes the condition that the requested resource must be available for the action to be applicable and that the resource will be occupied by the action during its application.

With the resource information, the first step for the inference engine is to identify critical sections of each subplan. A critical section is defined as a set of consecutive actions that must be executed as an indivisible planning step. When consecutive actions in a subplan declare the same resource, these actions form a critical section. Mutual exclusion between critical sections can be enforced by semaphors, as used for concurrency management in multiprocessing operating systems (Brinch Hansen [1973]). A semaphor is an integer variable shared by subplans; each resource is associated with one semaphor. The value of a semaphor, either zero or one, is used to signal the status of the resource. When the semaphor is one, the resource is available; when the semaphor is zero, the resource is

occupied. Using such a semaphor mechanism, a conflict-detection procedure based on resource reasoning can be implemented by a program module called "resource manager."

Plan Revision

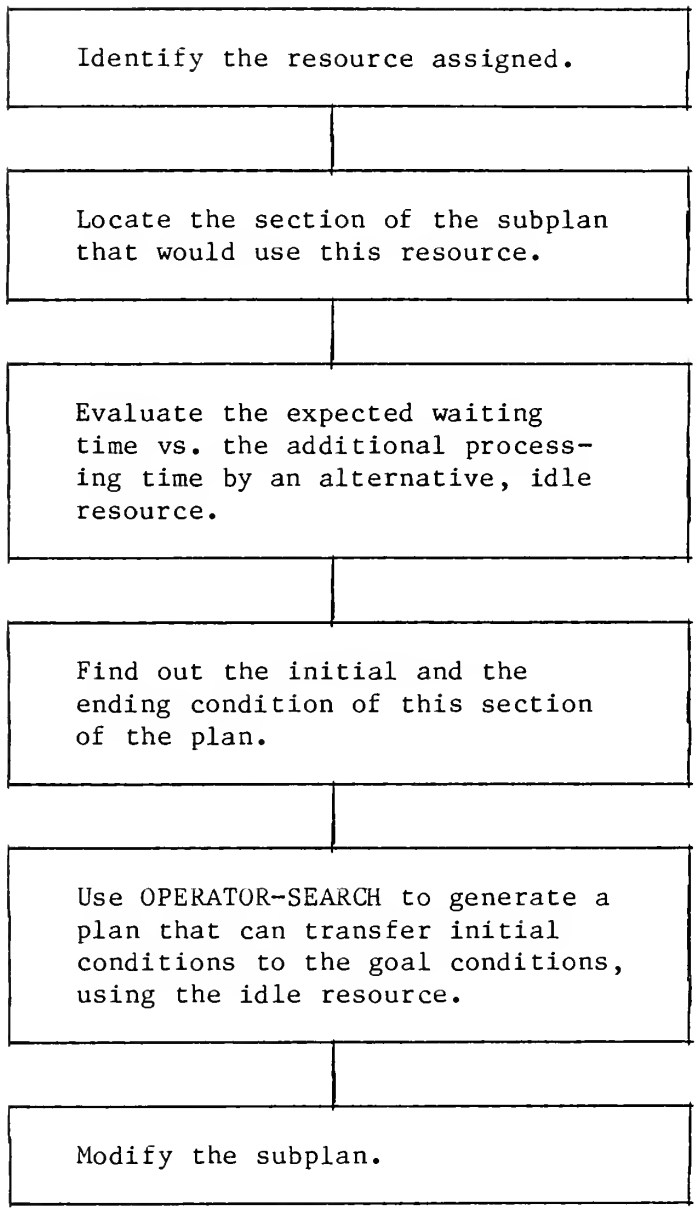
Because of the undeterministic nature of the environment, the planning and scheduling system for flexible manufacturing should be adaptable to unexpected events or system errors, such as:

- tool-breaks
- machine-malfunctions
- hardware errors
- long queue at bottlenecks
- workpiece collision
- programming logic errors
- misalligned fixtures

These are common events in any manufacturing systems, and usually can be corrected manually. But for flexible manufacturing with autonomous control, the planning and scheduling system, as the system's decision-maker, must take proper measure. One option is to interrupt and stop the operation when any of the unplanned events occur. As an alternative, we can use a PLAN-REVISION scheme to readjust the planning steps and bypass the troubled spot of the system. The flowchart for PLAN-REVISION is shown in Figure 6.

This approach displays some desirable characteristics for real-time planning and scheduling in the flexible manufacturing environment. First, it is goal-directed, i.e., users only need to specify the goals of the manufacturing process and the planning system would,

FIGURE 6


Identify the resource assigned.
Locate the section of the subplan that would use this resource.
Evaluate the expected waiting time vs. the additional processing time by an alternative, idle resource.
Find out the initial and the ending condition of this section of the plan.
Use OPERATOR-SEARCH to generate a plan that can transfer initial conditions to the goal conditions, using the idle resource.
Modify the subplan.

accordingly, derive the necessary steps on-line. Second, it is dynamically adjustable. New goals can be accommodated while the current production plan is still being executed; also, plans can be modified when unexpected events occur (e.g., tool or machine breakdowns). A plan-revision scheme is initiated when bottlenecks are detected; the scheme, in turn, seeks to use alternative resources to improve the throughput. Additional considerations should be given to the travel paths taken by guided carts or the arm movements of neighboring robots so that any potential conflicts or harmful interferences are avoided (Bourne and Fussell [1982], Lozana-Perez [1982]).

## V. SPECIAL SCHEDULING FUNCTIONS

### Dynamic Scheduling

In flexible manufacturing systems, frequently jobs arrive dynamically, each requiring a variety of operations. On the other hand, due to the regular advance customer orders and periodic maintenance, some job may be predetermined and scheduled. To schedule the unanticipated jobs in real-time and integrate them with the existing schedule for predetermined jobs is what we call dynamic scheduling.

Three schemes can be applied in achieving dynamic scheduling:

1. whenever an unanticipated order arrives, the scheduler regenerates the scheduling algorithm which re-schedules the remaining operations as well as the necessary operations for the newly arrived order;

2. the scheduler is activated periodically to schedule the jobs, including both static and dynamic jobs, that arrived after the last scheduling point; or

3. each time when an unanticipated job arrives, the scheduler only assigns unused resources to the new job.

The knowledge-based planning approach performs dynamic scheduling well because of its structured knowledge representations and the conflict-resolving inference capability. The current status of the cell, the progress of the ongoing plan, and the utilization of the machines are all included and updated in the world model. When new jobs need to be scheduled during the execution of existing jobs, a simple plan-modification procedure, such as the following one based on the foregoing first scheme, can be invoked to accommodate the new jobs.

Step 1. Establish plans for the new jobs based on the currently available machines.

Step 2. Use the conflict-resolution scheme to coordinate the actions for the new jobs and the remaining actions for the old jobs.

Step 3. Improve the modified plan by the same plan-revision scheme.

Because previously generated plans are stored in the data base with structured knowledge representation, new jobs change plans only where they interact with the new jobs. This concept originated in the STRIPS planning system, where macro-operators and structured plan-representation assisted both in the solution of similar problems and also in the intelligent monitoring of the plans' execution (Fike and Nilson [1971]). Dynamical adjustability, in short, is one of the primary advantages of the knowledge-based planning approach.

Distributed Scheduling

An emerging architecture for computer integrated manufacturing systems is the cellular system which consists of a collection of flexible manufacturing cells, each of which serves a specific part family. The planning and scheduling system discussed in this paper is

executed by each cell's host computer, its control unit. The planning problem for the whole CIMS is itself a two-level problem: the first-level planner distributes jobs among cells according to the capabilities and the set-up of each individual cell; the second-level planner--the one described in this paper--in turn performs the planning and scheduling within each cell. To achieve good performance, flexible manufacturing cells must coordinate and communicate with each other through a local area network. Shaw and Whinston [1983] analyzed distributed task allocation and modeled the first level planning problem by a distributed bidding algorithm. That paper developed a variant of the knowledge-based system that, guided by augmented Petri nets, acts as the first-level planner. Integration of the two planning systems can thus result in a distributed knowledge-based system, with the knowledge for both the job allocation and the inter-cell scheduling incorporated into the knowledge base in each cell.

## VI. CONCLUSION

In this paper, planning in the flexible manufacturing environment has been investigated in the context of a knowledge-based approach in order to handle the dynamically changing environment, interactions between manufacturing processes, and the versatility of processors. The nonlinear planning method developed in artificial intelligence has been extended so that the duration and resource information of each action is immediately derivable. The planning system can effectively schedule jobs in a flexible manufacturing cell and dynamically determine the routing of workpieces among the processors. The resulting

plan, a partially ordered network, demonstrates maximal parallelism and shortest duration.

REFERENCES

[1] Bourne, D. and Fussell, P., "Designing Programming Languages for Manufacturing Cells," The Robotics Institute, CMU-RI-TR-82-5, Carnegie-Mellon University, 1982.

[2] Brinch Hansen, P., Operating System Principles (Englewood Cliffs, New Jersey: Prentice-Hall), 1973.

[3] Bullers, W., Nof, S., and Whinston, A., Artificial Intelligence in Manufacturing and Control," AIIE Transactions, Volume 12, Dec. 1980, pp. 351-364.

[4] Chang, T. C. and Nau, D. S., "Prospects for Process Selection Using Artificial Intelligence," Computer in Industry, Vol. 4, No. 3, 1981.

[5] Cutkosky, et. al., "Precision Machining Cells within a Manufacturing System," The Robotics Institute, Carnegie-Mellon University, 1984.

[6] Davis, R., and King, J., "An Overview of Production Systems," in Machine Intelligence, E. Elcock and D. Michie (Eds.), Horwood, Chichester, England, 1977.

[7] Descotte, Y., and Latcome, J-C, "GARI: A Problem Solver that Plans How to Machine Mechanical Parts," Proceedings Seventh International Joint Conference on Artificial Intelligence, 1981, pp. 300-332.

[8] Fikes, R. E. and Nilson, N. J., "STRIPS: a new approach to the application of theorem proving to problem solving," Artificial Intelligence, 2 (3/4), (1971), pp. 189-208.

[9] Fikes, R. E., Hart, P. E. and Nilson, N. J., "Learning and executing generalized robot plans," Artificial Intelligence, 3(4), (1972), pp. 251-288.

[10] Fox, M. S., "The Intelligent Management System: An Overview," The Robotics Institute, Carnegie-Mellon University, 1982.

[11] Fox, M. S., Allen, B. P., Smith, S. F. and Strohm, G. A., "ISIS: A Constraint-Directed Reasoning Approach to Job Shop Scheduling," CMU-RI-TR-83-8, The Robotics Institute, Carnegie-Mellon University, 1983.

[12] Hutchingson, G. K., "Flexibility is Key to Economic Feasibility to Automating Small Batch Manufacturing," Industrial Engineering, (1984), pp. 77-84.

[13] Lozana-Perez, T., "Robot Programming," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, A.I. Memo No. 698, 1982.

[14] McDermott, J., "R1: A Rule-based Configurer of Computer Systems," Artificial Intelligence, Vol. 19, Sept. 1982, pp. 39-88.

[15] Merchang, M. E., "Production: A Dynamic Challenge," IEEE Spectrum, May 1983, pp. 36-39.

[16] Newell, A. and Simon, H., "GPS, A Program That Simulate Human Thought," in Computers and Thought, E. A. Feigenbaum and J. Feldman (Eds.), McGraw-Hill, New York, 1963.

[17] Nilsson, N., Principles of Artificial Intelligence (Palo Alto: Tioga), 1980.

[18] O'Connor, D., "Using Expert Systems to Manage Change and Complexity in Manufacturing," in Artificial Intelligence Applications for Business, W. Reitman (Ed.), Ablex Publishing Co., Norwood, New Jersey, 1984

[19] Sacerdoti, E. D., A Structure for Plans and Behavior (New York: North-Holland), 1977.

[20] Shaw, M. J. P. and Whinston, A. B., "Automatic planning and flexible scheduling: A knowledge-based approach," Proceedings International Conference on Automation and Robotics, St. Louis, 1985.

[21] Shaw, M. J. P. and Whinston, A. B., "A Knowledge-Based Approach to Planning and Scheduling in Flexible Cells," Technical Report, Department of Business Administration, University of Illinois, 1985.

[22] Shaw, M. J. P., The Design of a Distributed Knowledge-Based System for Intelligent Manufacturing Information Systems, Ph.D. Thesis, Purdue University, 1984.

[23] Shaw, M. J. P. and Whinston, A. B., "Distributed planning in cellular flexible manufacturing systems," Management Information Research Center, Purdue University, 1983.

[24] Shortliffe, E., Computer-based Medical Consultations: MYCIN, Elsevier, New York, 1976.

[25] Tate, A., "Generating project networks," Proceedings International Joint Conference on Artificial Intelligence, 1977.

[26] Vere, S., "Planning in time: windows and durations for activities and goals," Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 3, (1983), pp. 246-266.

[27] Wilkins, D., "Parallelism in planning and problem solving: reasoning about resources," Tech. Note 258, AI Center, SRI, 1982.

[28] Wilkins, D. E., "Domain independent planning: representation and plan generation," Artificial Intelligence, 22, (1984), pp. 269-301.

## APPENDIX

Shaw and Whinston [1985] show the application of the nonlinear planning approach to the scheduling problem in a three-machine manufacturing cell. To simplify the situation, suppose there are two independent jobs to be scheduled. The first job, requiring operations OP1, OP2, and OP3, is assigned the following routing by step 1 of nonlinear plan construction-linear planning:

p1 ENTER

p2 EXECUTE(LOAD,DOCK)

p3 TRANSFER(DOCK,M1)

p4 EXECUTE(M1,OP1)

p5 TRANSFER(M1,M2)

p6 EXECUTE(M2,OP2)

p7 TRANSFER(M2,M3)

p8 EXECUTE(M3,OP3)

p9 UNLOAD(M3,DOCK)

p10 EXIT

Similarly, job 2, requiring operations OP1, OP3, will be assigned the following linear routing

q1 ENTER

q2 EXECUTE(LOAD,DOCK)

q3 TRANSFER(DOCK,M1)

q4 EXECUTE(M1,OP1)

q5 TRANSFER(M1,M3)

q6 EXECUTE(M3,OP3)

q7 UNLOAD(M3,DOCK)

q8 EXIT

A portion of the planning steps generated by PLAN-AHEAD is shown in Figure A.1. The resulting schedule and the corresponding machine loading is depicted in Figure 3(a). After plan revision, the seche-dule is improved to the one shown in Figure 3(b). The final schedule can be represented by a partially ordered network, as shown in Figure A.2.

FIGURE A.1

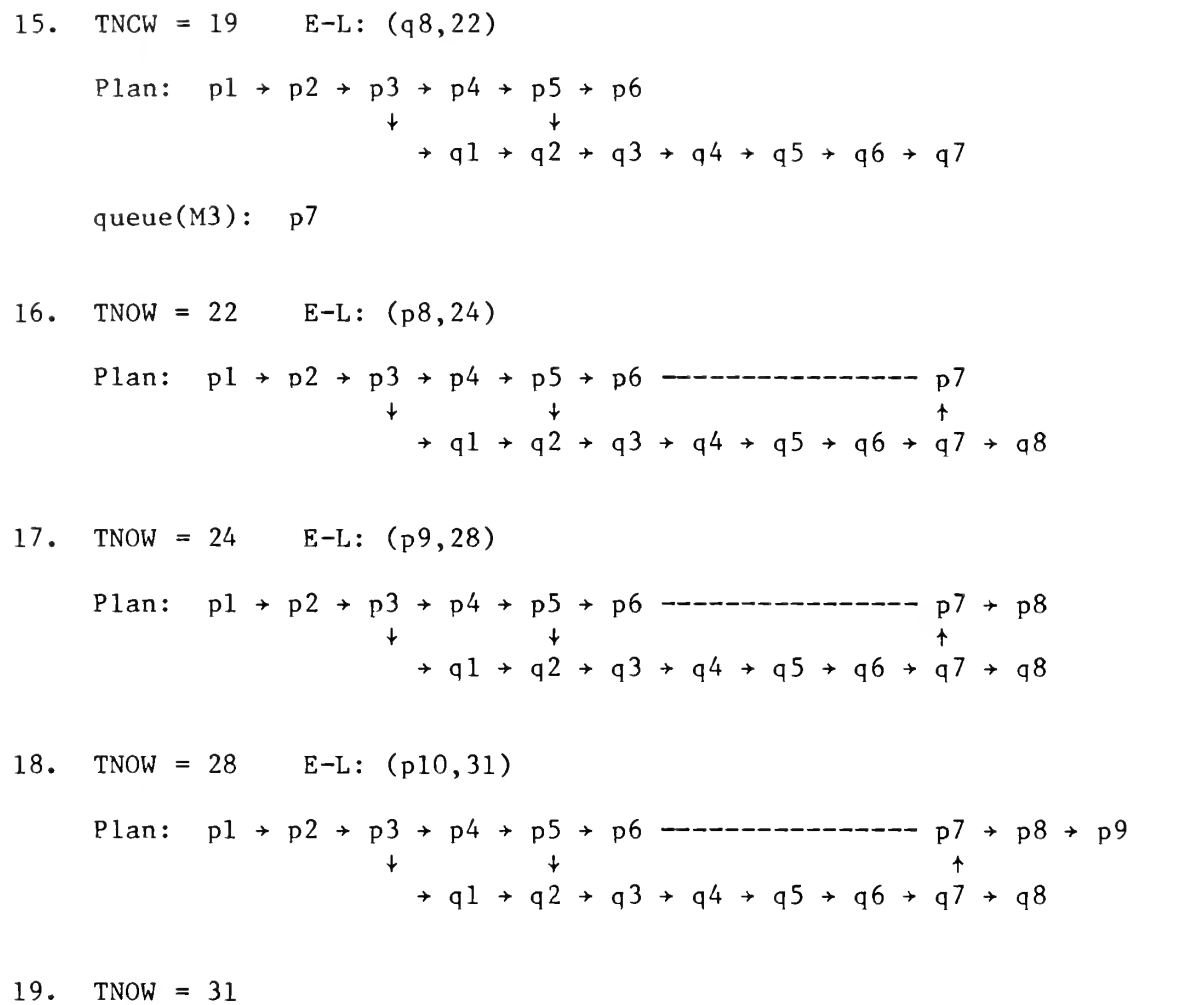

15. TNCW = 19 E-L: (q8,22)
Plan: p1 → p2 → p3 → p4 → p5 → p6
→ q1 → q2 → q3 → q4 → q5 → q6 → q7
queue(M3): p7
16. TNOW = 22 E-L: (p8,24)
Plan: p1 → p2 → p3 → p4 → p5 → p6 ---------- p7
→ q1 → q2 → q3 → q4 → q5 → q6 → q7 → q8
17. TNOW = 24 E-L: (p9,28)
Plan: p1 → p2 → p3 → p4 → p5 → p6 ---------- p7 → p8
→ q1 → q2 → q3 → q4 → q5 → q6 → q7 → q8
18. TNOW = 28 E-L: (p10,31)
Plan: p1 → p2 → p3 → p4 → p5 → p6 ---------- p7 → p8 → p9
→ q1 → q2 → q3 → q4 → q5 → q6 → q7 → q8
19. TNOW = 31
Plan: p1 → p2 → p3 → p4 → p5 → p6 ---------- p7 → p8 → p9 → p1
→ q1 → q2 → q3 → q4 → q5 → q6 → q7 → q8

FIGURE A.2